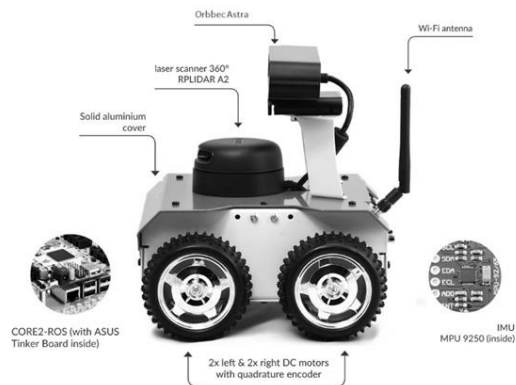


Word Count:
1950

Year: 2021

ROSBOT 2.0 PRO



CE315 – Mobile Robotics Assignment 1

[1603905]

JONATHAN DEVAUX

Table of Contents

Introduction:	2
Implementation of Task 1:	4
Flowchart:	4
Implementation of Task 2:	5
Implementation of Task 3:	7
Flowchart:	7
Environment Map:	8
Appendix:	9
Task 1:	9
Task 2(A):.....	9
Task 2(B):.....	10
Task 3:	11

Introduction:

There are various types of autonomous robots used in real world applications for example autonomous machinery that builds cars and puts parts together. A sector that uses autonomous robots a lot is the manufacturing sector as this can be seen in factories, an example of this tea factory that takes in the materials which the machines separates tea material and puts them into tea bags for suppliers to create the product. The criteria for robots to be autonomous robots is that they need to be autonomous, adaptive, robust, modularity, communication, and high bandwidth to achieve high performance to be effective.

Humanity needs mobile robots to simplify things and assist with tasks so that society has more free time innovate and develop new ways of finding cleaner technology that leads to rapid development for society. Also, autonomous robots replace people from working in hostile environments which this can save lives and is economically cheaper in the long run. However, without mobile robots society will eventually stagnate and fall apart due to the number of processes mobile robots are involved in nowadays.

Below is a list of autonomous robots used in real world applications:

- **Autonomous robots for Logistics**
 - This means that robots can transport orders across the warehouse or through a facility many times compared to using human workers who would require breaks as this is a labour intensive role where human workers can be used for other vital jobs.
- **Autonomous robots for E-Commerce**
 - There are many forms of this such as moving carts and being used for tasks such as order fulfilment, returns handling, parcel sortation, inventory managements(resource allocation) and raw materials transport & sorting.
- **Autonomous robots for Warehousing/Distribution Centres**
 - Used for doing the heavy lifting and transport throughout the warehouse which this frees human workers from spending time travelling through the warehouse.
 - The robots have the ability to see and localise in open spaces and use the laser detection as a sensor.
- **Autonomous robots for Data Centres**
 - Used in data centres and research facilities which can transport high value materials safely. Therefore, this enables instant and accurate accessible documentation of the process easily.
- **Autonomous robots in Healthcare**
 - Autonomous robots are used to transport supplies and medicine throughout the healthcare facility. An important use of them is in infectious disease units which prevents doctors and nurses from infection risk whilst treating the patient, e.g., robot nurses.
 - Can be used in sanitation if the robots are fitted with UV lights/decontamination sprays to kill viruses/germs. (An ongoing example is the current Coronavirus pandemic)

- **Autonomous robots in Biotech**
 - Combined with robotic arms these robots can be used to control the regulated production processes such as process inputs, regular monitoring tasks as well waste removal from the production line.
 - An example of this is the recent ongoing Coronavirus pandemic where biopharmaceuticals have to store the Coronavirus vaccine in temperatures below zero degrees at a certain temperature to prevent the vaccines from becoming denatured/spoiled which the autonomous robot will control the temperature of the stored vaccines.
- **Autonomous robots for Military**
 - Earliest example of autonomous robots used was in World War two were the V1 & V2 rockets. It features autopilot and automatic detonation, these were the predecessors to modern cruise missiles
 - Autonomous robots for military use has been used in bomb disposal, evidence of this is from the early 2000's onwards when the US military started using robots for bomb disposal both in Iraq and Afghanistan to clear landmines and improvised bombs. This have saved soldiers lives who otherwise may loose their lives doing the bomb disposal themselves or treading on a landmine.
 - Another example of this is the use autonomous drones and unmanned aircraft/aerial vehicle(UAV) which this has caused ethical debate due to the possibility of the drones being hacked. However, they would follow the mission parameters.
 - In addition to this there has been recent talk of British army using robots for infantry by 2030 which the link is here: ([British Army chief: A quarter of soldiers could be robots by 2030 \(iiottechnews.com\)](https://www.iiottechnews.com))
- **Autonomous robots for Space**
 - Autonomous robots have been used for space exploration for decades, an example of this is when NASA sent two unmanned spacecraft in the Viking program to Mars in 1976.
 - Further space exploration will require the use of Autonomous robots to explore nearby planets and systems due to the current technological limitations and because sending astronauts would take too long and the number of resources as well as expense would be too great of factor for any country or space agency to afford.
 - In addition, autonomous robots such as the Mars rover can survive in harsh environments whether it is sub-zero temperatures or scorching heat, and they are designed to work in limited oxygen environments
 - Android Astronauts these robots help astronauts with assisting operations such as maintenance of the spacecraft.
- **Autonomous robots for Transportation**
 - An example of this is the Docklands Light Railway(DLR) as there is no human driver and instead is autonomous driving.
 - In recent times Tesla and a few other car companies have been testing out autonomous cars which if this comes into fruition then this could reduce traffic and pollution.

Implementation of Task 1:

Flowchart:

Flowchart key:

Start/Finish



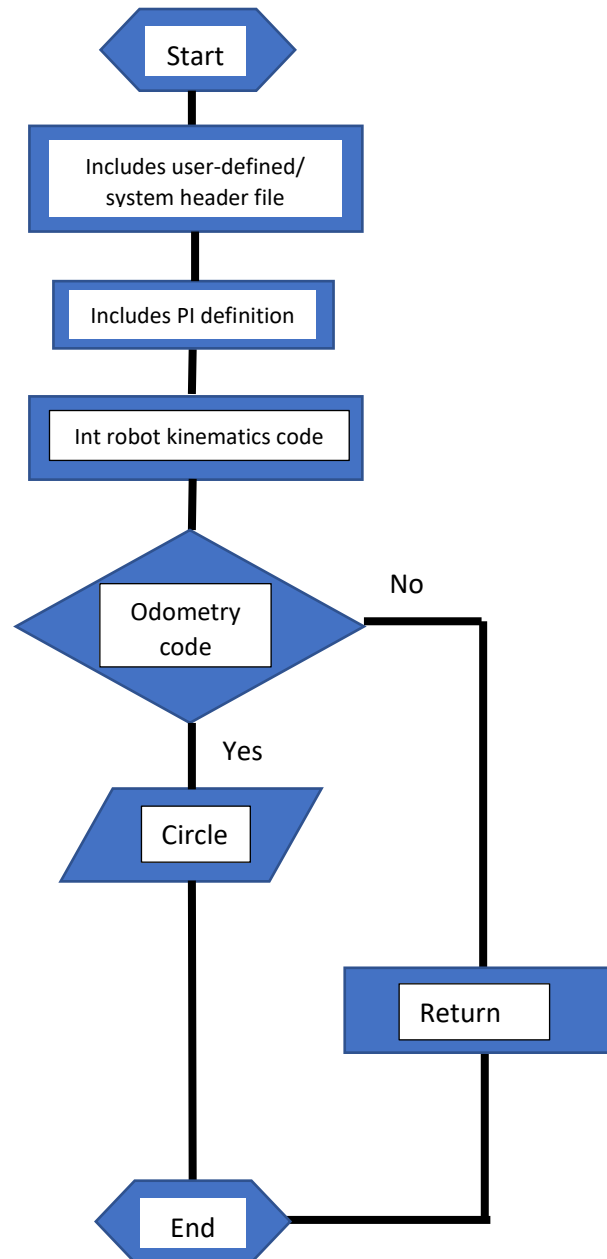
Computation/
processing



Comparison



Input/Output



Implementation of Task 2:

In task 2 the results were saved into a text file to be later plotted into the scatter graphs, two text files were created to represent the two different velocity parameters as well as two separate data files to create the two scatter graphs which are figure 1 and figure 2.

The scatter graphs below are to generate the robot trajectory when $K=0,1,2,3,4,5,6,7\dots 200$ (200 results). The result of this is when the vehicle left velocity is equal to 10cm/s and the vehicle right velocity is equal to 8cm/s demonstrated in figure 1. The shape this represents is an eclipse shape.

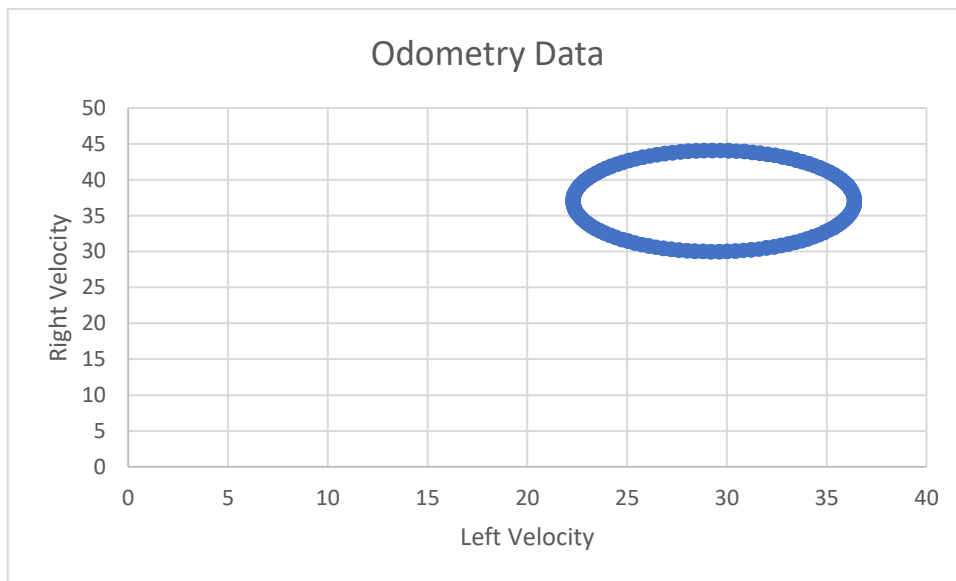


Figure 1

In the second scatter graph below (figure 2) the parameters for both the vehicle left velocity and right velocity have changed which vehicle left velocity is equal to 5cm/s and vehicle right velocity is equal to 7cm/s in figure 2. The shape of the scatter graph represents a larger eclipse shape compared to the eclipse in figure 1 above.

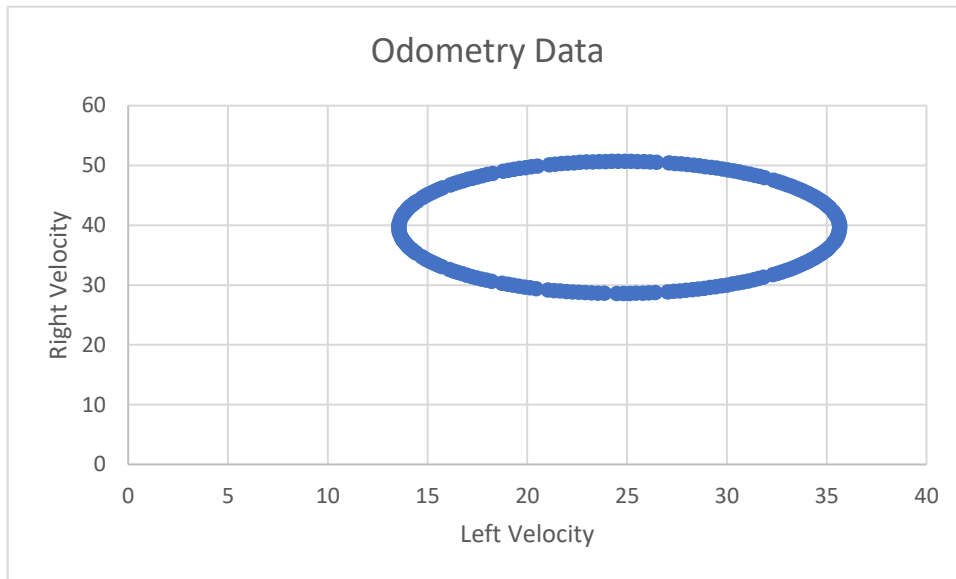


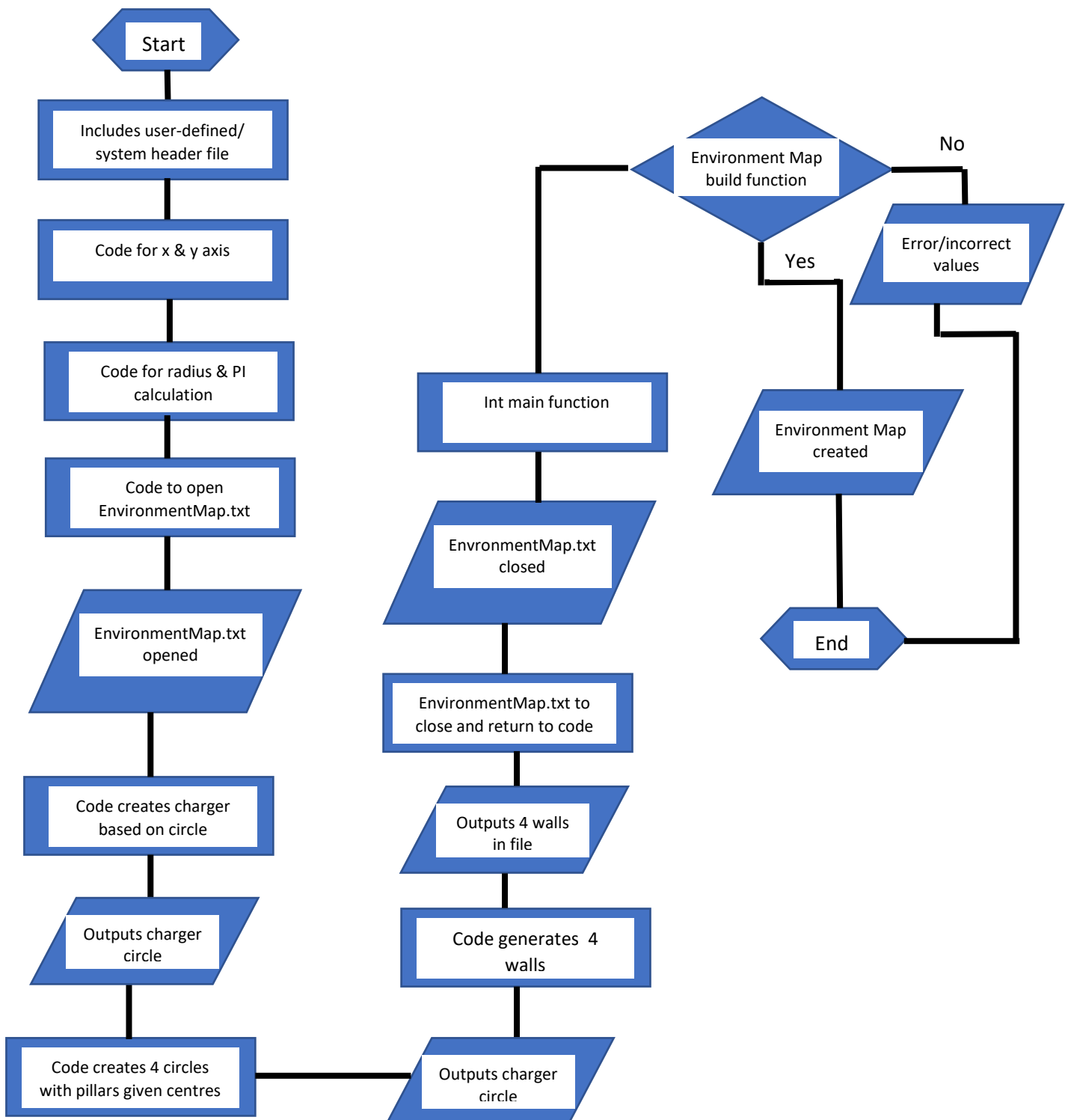
Figure 2

In addition to this, both scatter graphs have an eclipse shape and what can be taken from this is when the velocity is reduced the eclipse shape will be larger due to the Rosbot speed and this can be seen in figure 2. However, when the velocity for left and right is increased the eclipse shape is smaller due to the velocity change and this is demonstrated in figure 1. This means the Rosbot wheels are turning faster as both left and right velocities have different velocity instead of the same for both right and left as that would mean the Rosbot would not turn but move in a straight line.

Implementation of Task 3:

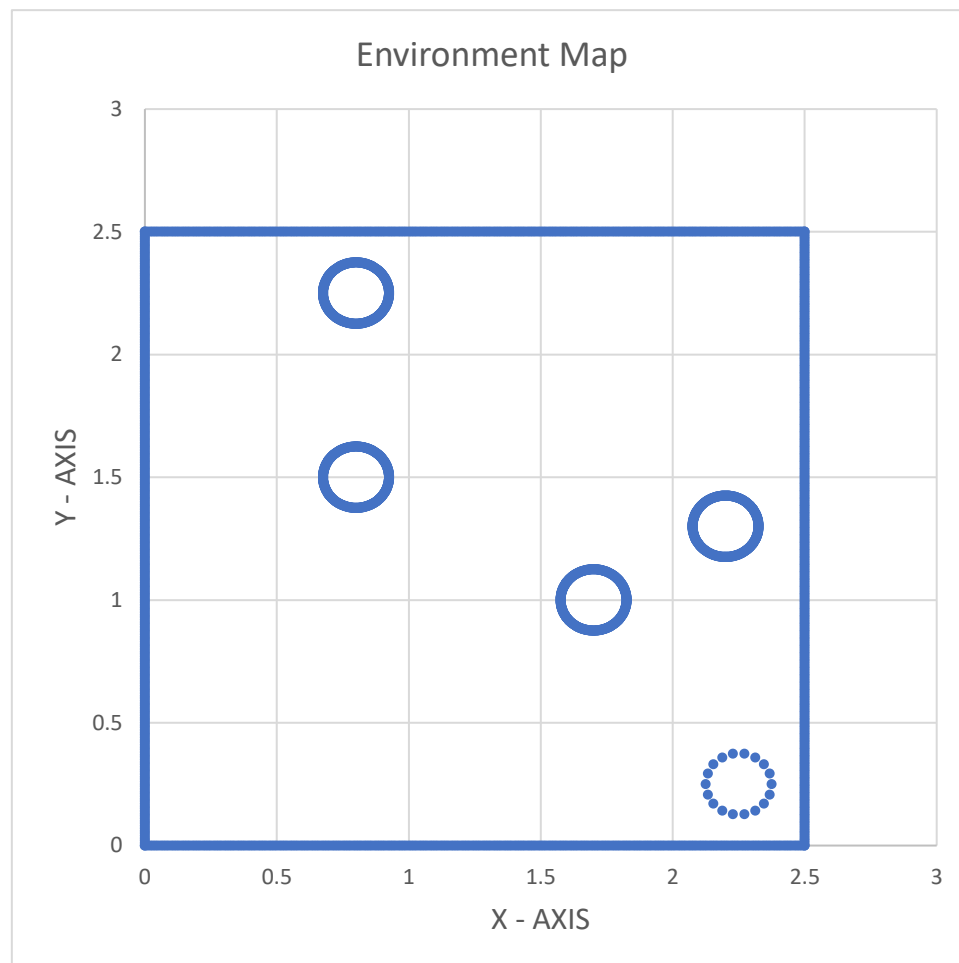
The flowchart for task 3 below uses the same flowchart key as displayed in page 3 that is used in the flowchart for task 1. As can be seen in the flowchart a lot of computation/processing was used in the code compared to comparisons this meant that output can be more relevant and accurate.

Flowchart:



Environment Map:

The environment map displayed below features four walls in square formation which 1 wall represents a line for example in coordinates (0 , 2.5). The charging point is represented as a circle in the coordinates block within (2 , 0.5) whilst the other circles are represented as filled circles like the circle in the coordinates block within (0.5 , 2.5) are merely obstacles that the robot will have to avoid once the environment map has been fully developed for the robot to traverse.



Additionally, the Rosbot will have to traverse through the circles(obstacles) and not around the obstacles to reach the charging point successfully with the accurate parameters set.

Appendix:

Here below is the code used for each task comments are displayed using “//” and are in green. For task 1 the calculations used for odometry calculation was used in task 2. In regards to task 2 it is split into 2 programs which the headings are labelled task 2(A) and task 2(B) both programs contain different velocities. In task 3 the environment map is created and commented on.

Task 1:

```
//Task 1
#include<math.h>
#include<iostream>
#include<fstream>

int robot_kinematics(double left_vel, double right_vel) //left velocity & right
velocity
{
    int i;
    for(i=0; i<SIZE; i++)
    {
        rob_x[i+1]=rob_x[i]+(left_vel + right_vel)/ 2*cos(rob_theta[i])*delta_t;
        //two codes for the equationshere
        rob_y[i+1]=rob_y[i]+(left_vel + right_vel)/2*sin(rob_theta[i])*delta_t;
        rob_theta[i+1]=rob_theta[i]+(right_vel - left_vel/wheelbase)*delta_t;
    }
    return 0;
}
```

Task 2(A):

```
//Task 2

#include<math.h>
#include<iostream>
#include<fstream>

#define PI 3.14159265
int wheelbase=30, delta_t=1;
const int SIZE=200; //k = 1,2,3,4,5,6,7....200 (this means 200 results)
double vl=10,vr=8; //paremeters given
double rob_x[SIZE]={30}, rob_y[SIZE]={30}, rob_theta[SIZE]={PI/4};

int robot_kinematics(double left_vel, double right_vel) //left velocity & right
velocity
{
    int i;
    for(i=0; i<SIZE; i++)
    {
        rob_x[i+1]=rob_x[i]+(left_vel + right_vel)/ 2*cos(rob_theta[i])*delta_t;
        //two codes for the equationshere
        rob_y[i+1]=rob_y[i]+(left_vel + right_vel)/2*sin(rob_theta[i])*delta_t;
        rob_theta[i+1]=rob_theta[i]+(right_vel - left_vel/wheelbase)*delta_t;
    }
    return 0;
}
```

```

FILE *fp;
int main(int argc, char **argv)
{
    int i;
    fp = fopen("odometry_data1", "w"); //opens odometry data as writable file
    robot_kinematics(vl, vr);
    // open a file for recording robot_kinematics(vl, vr);

    fprintf(fp, "%d, %d \n", 30, 30); //records the trajectory data
    for(i=1; i<SIZE; i++){
        fprintf(fp, "%f, %f\n", rob_x[i], rob_y[i]);
    } // record initial positions
    fclose(fp);
    return 0;
}

```

Task 2(B):

```

//Task 2

#include<math.h>
#include<iostream>
#include<fstream>

#define PI 3.14159265
int wheelbase=30, delta_t=1;
const int SIZE=200; //k = 1,2,3,4,5,6,7....200 (this means 200 results)
double vl=5, vr=7; //parameters given
double rob_x[SIZE]={30}, rob_y[SIZE]={30}, rob_theta[SIZE]={PI/4};

int robot_kinematics(double left_vel, double right_vel) //left velocity & right
velocity
{
    int i;
    for(i=0; i<SIZE; i++)
    {
        rob_x[i+1]=rob_x[i]+(left_vel + right_vel)/ 2*cos(rob_theta[i])*delta_t;
        //two codes for the equationshere
        rob_y[i+1]=rob_y[i]+(left_vel + right_vel)/2*sin(rob_theta[i])*delta_t;
        rob_theta[i+1]=rob_theta[i]+(right_vel - left_vel/wheelbase)*delta_t;
    }
    return 0;
}

//T2
FILE *fp;
int main(int argc, char **argv)
{
    int i;
    fp = fopen("odometry_data", "w"); //opens odometry data as writable file
    robot_kinematics(vl, vr);
    // open a file for recording robot_kinematics(vl, vr);

    fprintf(fp, "%d, %d \n", 30, 30); //records the trajectory data
    for(i=1; i<SIZE; i++){
        fprintf(fp, "%f, %f\n", rob_x[i], rob_y[i]);
    } // record initial positions
    fclose(fp);
    return 0;
}

```

Task 3:

```
#include<iostream>
#include<fstream>
#include<math.h>
using namespace std;

#define PI 3.14159265
int build_environment_map(){
// To initialize all the parameters by using data in the figure above.

double x0[5]={0.8, 0.8, 1.7, 2.2, 2.25}; //x axis points
double y0[5]={1.5, 2.25, 1.0, 1.3, 0.25}; //y axis points
double radius=0.125, wallLength=2.5;
double RD=PI/180;
// Opens a data file called "EnvironmentMap.txt" for storing walls, pillars and
charger generated by the code.

ofstream map;
map.open("EnvironmentMap.txt");

// Creates a charger based on its centre.
for (int i=0; i<360; i=i+20) {
map<<x0[4]+radius*cos(i*RD)<<' '<<y0[4]+radius*sin(i*RD)<<endl;
}

// Generates 4 circle pillars with the given centres.
for (int i=0; i<360; i++) {
map<<x0[0]+radius*cos(i*RD)<<' '<<y0[0]+radius*sin(i*RD)<<endl;
}
for (int i=0; i<360; i++) {
map<<x0[1]+radius*cos(i*RD)<<' '<<y0[1]+radius*sin(i*RD)<<endl;
}
for (int i=0; i<360; i++) {
map<<x0[2]+radius*cos(i*RD)<<' '<<y0[2]+radius*sin(i*RD)<<endl;
}
for (int i=0; i<360; i++) {
map<<x0[3]+radius*cos(i*RD)<<' '<<y0[3]+radius*sin(i*RD)<<endl;
}

// Used to generate 2 walls.
for(double j=0 ; j<wallLength; j=j+0.01){
map << j <<' '<< 0 << endl;}
for(double j=0 ; j<wallLength; j=j+0.01){
map << wallLength <<' '<< j << endl;}
// code below generates the other 2 walls
for(double j=0 ; wallLength>j; j=j+0.01){
map << j <<' '<< 2.5 << endl;}
for(double j=0 ; wallLength>j; j=j+0.01){
map << 0 <<' '<< j << endl;}

map.close();
return 0;
}

// Main function in code for compilation
int main(int argc, char **argv)
{
int i;
i = build_environment_map();
return 0;
}
```